

# Fast Track Exercises to Understand ARM Cortex-M4 Architecture Using Texas Instruments' Stellaris Launch Pad

Rajeswari Cherukuri, Raghavendra Rao Kanchi\*

VLSI and Embedded System Laboratory, Department of Physics, Sri Krishnadevaraya University, Anantapuramu, India

## Email address:

shahitharaji@gmail.com (R. Cherukuri), kanchiraghavendrarao@gmail.com (R. R. Kanchi)

## To cite this article:

Rajeswari Cherukuri, Raghavendra Rao Kanchi. Fast Track Exercises to Understand ARM Cortex-M4 Architecture Using Texas Instruments' Stellaris Launch Pad. *American Journal of Embedded Systems and Applications*. Vol. 3, No. 3, 2015, pp. 23-34.

doi: 10.11648/j.ajes.20150303.11

---

**Abstract:** In this paper we present fast track training for students to get acquainted with ARM Cortex -M4 architecture. This is accomplished by considering exercises and a simple project that gives not only an insight but also hands-on experience to the students and electronics hobbyist. Microcontrollers with ARM architecture have become popular and are one of the best microcontrollers to start working with in an introductory embedded system laboratory. Texas Instruments' Stellaris LaunchPad is chosen for this purpose. The tiny board contains LM4F120H5QR as the microcontroller. It is an inexpensive board which can be purchased by the student. Further, these exercises with some more advanced projects can be introduced as a half-semester laboratory training program.

**Keywords:** Embedded System Laboratory, LM4F120H5QR Microcontroller, Project-Based Exercises

---

## 1. Introduction

ARM core microcontrollers have become popular and are being used in a wide range of applications. Reasons for becoming popular include its ability to execute instructions at a faster rate (few hundreds of MHz), pipelining execution, floating point arithmetic support for digital signal processing (DSP) applications, memory protection, and ultra low power consumption with a better interrupt support, etc. ARM cortex architecture supports 16-bit thumb and 32-bit mode applications. These features are further augmented with powerful on-chip peripherals. Present day microcontrollers with system-on-chip (SoC) contain a CPU with ARM cortex architecture plus Wi-Fi Network Processor Subsystem for Internet of things (IoT) applications developed on single silicon. Some of the examples for such microcontrollers are: Texas Instruments' CC3200, Nordic Semiconductors' nRF5182 and ST Microelectronics' STM32W108CB.

It is evident from the above facts that there is a fast growth and improvement in the field of semiconductor technology/electronic industry. This fact need to be backed up with modified curriculum at the undergraduate level, since the upcoming students are the workforce in the industrial environment. We propose in this paper, exercises and simple student project for undergraduate curriculum which fits to a

half-a-semester hands-on laboratory.

General areas that employ embedded systems covers every branch of day to day science and technology, namely communications, automotive, medical, consumer, machine control etc. Another trend in embedded device development is to produce families of devices with a basic functionalities and different extra details determining peculiarities of the device exploitation and finally the cost of the device. With embedded systems becoming omnipresent, there is a growing need to teach and train engineers to learn their design and development [1].

Learning embedded system is interesting on one hand but need efforts to learn many areas: microcontroller architecture and programming, sensor interfacing technology etc. Designing an embedded system laboratory is an art, and demands talented students to take up the design challenges keeping the time frame in their mind [2, 3]. Various methods, means and difficulties in establishing an embedded system design was explained by Li Tu and Jun Yang [4]. In our laboratory students are first trained to exploit the on-chip facilities of the microcontroller before developing any dedicated hardware. In this direction, we have published papers related to laboratory training to get hands-on experience and also as take-home exercises [5-8].

The Stellaris LM4F120 LaunchPad is a low-cost evaluation board from Texas Instruments [9]. It is built around ARM

Cortex M4-F based microcontroller. This processor provides multiple interfaces using AMBA<sup>R</sup> technology. It provides high-speed, low-latency memory access. Further, the processor has a memory protection unit to enable memory control and security privilege levels. The memory is organized in such a way that code, data and stack are served exclusively in a manner of task-by-task. When the on-chip peripherals of the microprocessor unit is looked at, it has a variety of peripherals (section 2). To understand the functionality of these peripherals for interfacing, we explained a series of exercises related to this. By performing exercises related to module A, the student will have an understanding of the Input/Output interfacing of LEDs, push button, seven segment displays, LCD module and mechanical device like stepper motor. Module B gives an opportunity to exploit the on-chip peripherals like: analog-to-digital converter (ADC), Universal Asynchronous Receiver- Transmitter (UART), Pulse Width Modulation (PWM), Inter- Integrated Circuit (I2C). Spoon feeding is avoided by leaving certain on-chip peripherals: Synchronous Serial Interface (SSI), Controller Area Network (CAN), Universal Serial Bus (USB) controller, analog comparator unexplained. The student can have a go, to understand these peripherals by developing suitable hardware and software, by which partial assessment of a student is made. At the end semester, students are supplied with a questionnaire to give their feedback. Results of this are included at the end of the paper (Section-5), which shows that students found it more convenient to use the Stellaris LaunchPad in getting hands-on training on exploiting the on-chip peripherals of ARM Cortex-M4 architecture in general and the on-board facilities provided by LaunchPad in particular.

The exercises are described in the order of increasing complexity, starting with a LED blinking (Hello World!) to temperature measurement by interfacing a sensor to the LaunchPad. The systematic approach by learn-while-doing, not only increases confidence in the student, but light up the spark of innovation by thinking new ways of using the microcontroller.

The paper is organized as follows: Section 2 gives brief description on the architectural details of LM4F120H5QR microcontroller. The programming procedure of the controller using Code Composer Studio (CCS) is described in Section 3. Section 4 gives the hardware and software details of the experiments developed in the present study. Section 5 gives the student feedback and survey. Conclusions are included in Section 6.

## 2. Architecture Description of LM4F120H5QR Controller

The TI Stellaris LM4F120H5QR microcontroller is based on an 80MHz version of the ARM Cortex-M4F processor core. The ARM Cortex-M4F processor core is a full-fledged 32-bit processor core and the "F" designates the inclusion of a hardware floating point unit. The TI LM4F120H5QR microcontroller also incorporates 256Kbytes of Flash

EPROM, 32Kbytes of SRAM, and 2Kbytes of EEPROM for memory [9].

Key features of the LM4F120H5QR microcontrollers include:

- IEEE754-compliant, single-precision floating-point capability at 80 MHz
- SIMD instructions
- 256Kbytes of embedded flash memory and 32Kbytes of SRAM
- Low-power modes including power-saving hibernate
- 32-bit ARM cortex-M4F architecture optimized for small-footprint embedded applications.
- Thumb-2 mixed 16-/32-bit instruction set
- Harvard architecture characterized by separate buses for instruction and data.
- Efficient processor core, system and memories
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionally
- Serial wire Debug and serial wire trace reduce the number of pins required for debugging and tracing
- Ultra-low power consumption with integrated sleep modes
- LM4F120H5QR microcontroller operates at 3.3V. In active mode controller uses 50mA at 3.3V and in idle mode it uses 12mA at 3.3V
- Two 12-bit 1MSPS ADCs and 24 analog input channels
- Two CAN controllers
- Optional full-speed USB 2.0 with device, host, and OTG
- Advanced motion control capability, with as many as 16 motion control PWM outputs
- Eight UARTs, six I2C modules, and four SPI/SSI ports, JTAG and ARM serial wire Debug facility is provided.

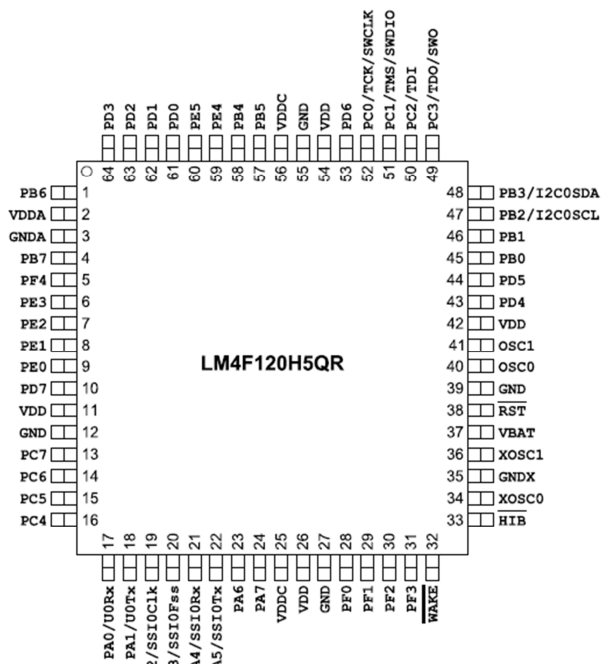
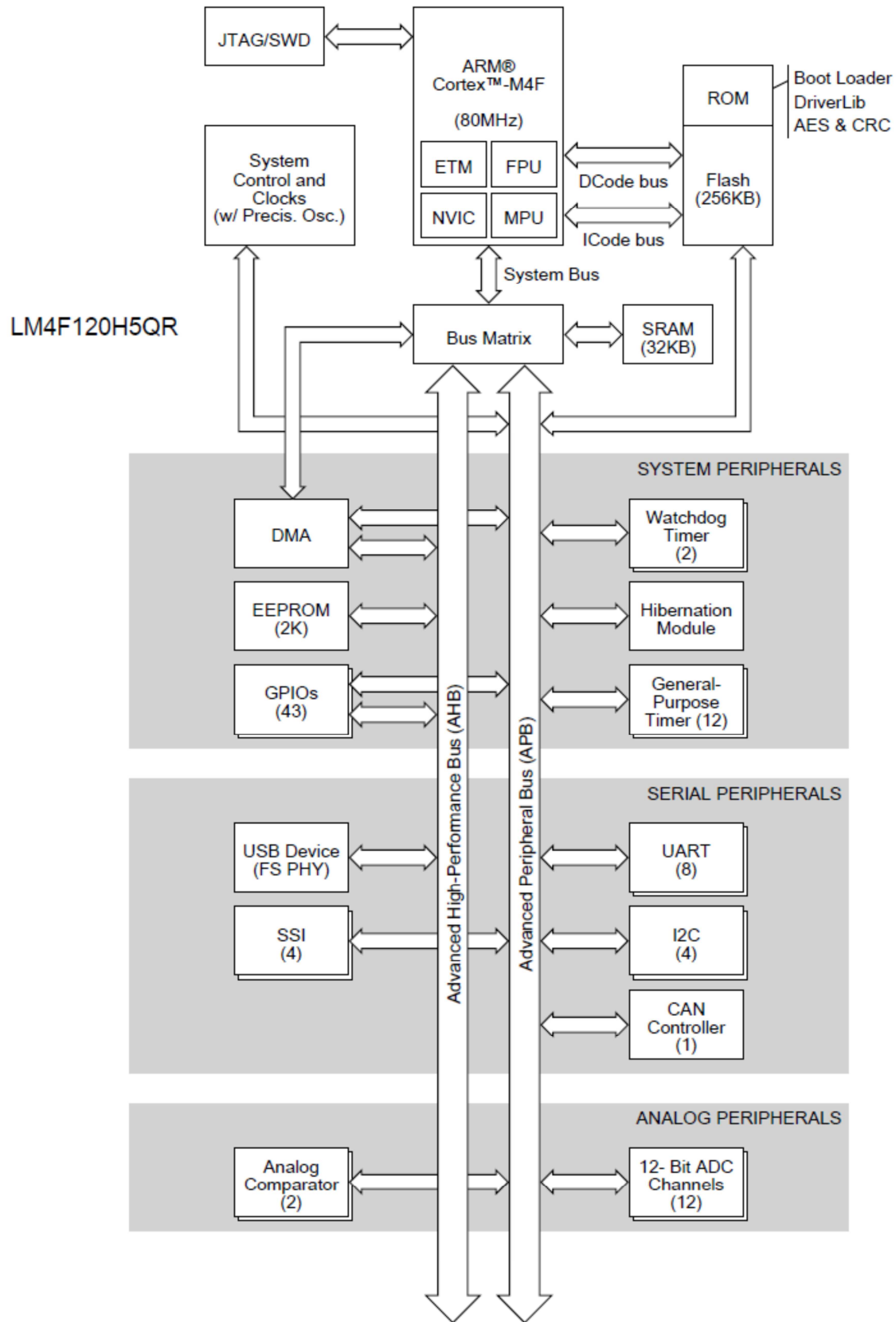


Figure 1. The pin diagram of LM4F120H5R microcontroller (courtesy: Texas Instruments, USA).

Texas Instruments user manual gives a detailed description on the market [11].  
on LM4F120H5QR [10]. Also, there is a text book available



**Figure 2.** Block diagram of LM4F120H5QR microcontroller (courtesy: Texas Instruments, USA).

### 3. Programming Procedure [12]

The main aim of using microcontrollers is to use it in dedicated or stand-alone applications. As such, the software developed for such application has to be stored in the microcontroller's application memory permanently. This method of storing is otherwise known as programming. The LM4F120H5QR microcontroller can be programmed using one of the two programming hardware options which are given below.

- Using Joint Test Action Group (JTAG)
- Using Serial Wire Debug (SWD)

In the present work, the on-board microcontroller is programmed using JTAG (USB Emulator connector). The application program can be developed either in assembly or embedded C. We used CCS integrated development environment (IDE) in the present work.

In order to give the flavor of CCS, the example of Blink program is considered.

After installing the CCS software, the shortcut icon on desktop can be clicked. A new project can be selected from the

drop-down menu, by giving project name, file name and the location. The project can be saved in the StellarisWare workspace. Further the device to be programmed and the in circuit debug (ICD) interface are to be selected. Now, an empty project space with the filename with .C extension will be created. The program to be run is entered in the file, and after completion the file can be saved with a name, e.g., LED\_Blink. C if the program is developed in C and with an extension of .asm if the program is developed in assembly language. Next in the project explorer window, the LED Blink project with its properties can be selected. The properties include path and inclusion of appropriate library files. Pressing OK button will finish the project explorer. Now the current project can be built. After building the project without errors, the next step is debugging. After successful debugging, pressing of Run button dumps the .HEX file on to the microcontroller. Now, the microcontroller can be run in the stand-alone mode for that specific application for which the software is developed. Some screenshots which are related to create a new project, built, debug, and run the project are shown in figures 3, 4, and 5 respectively.

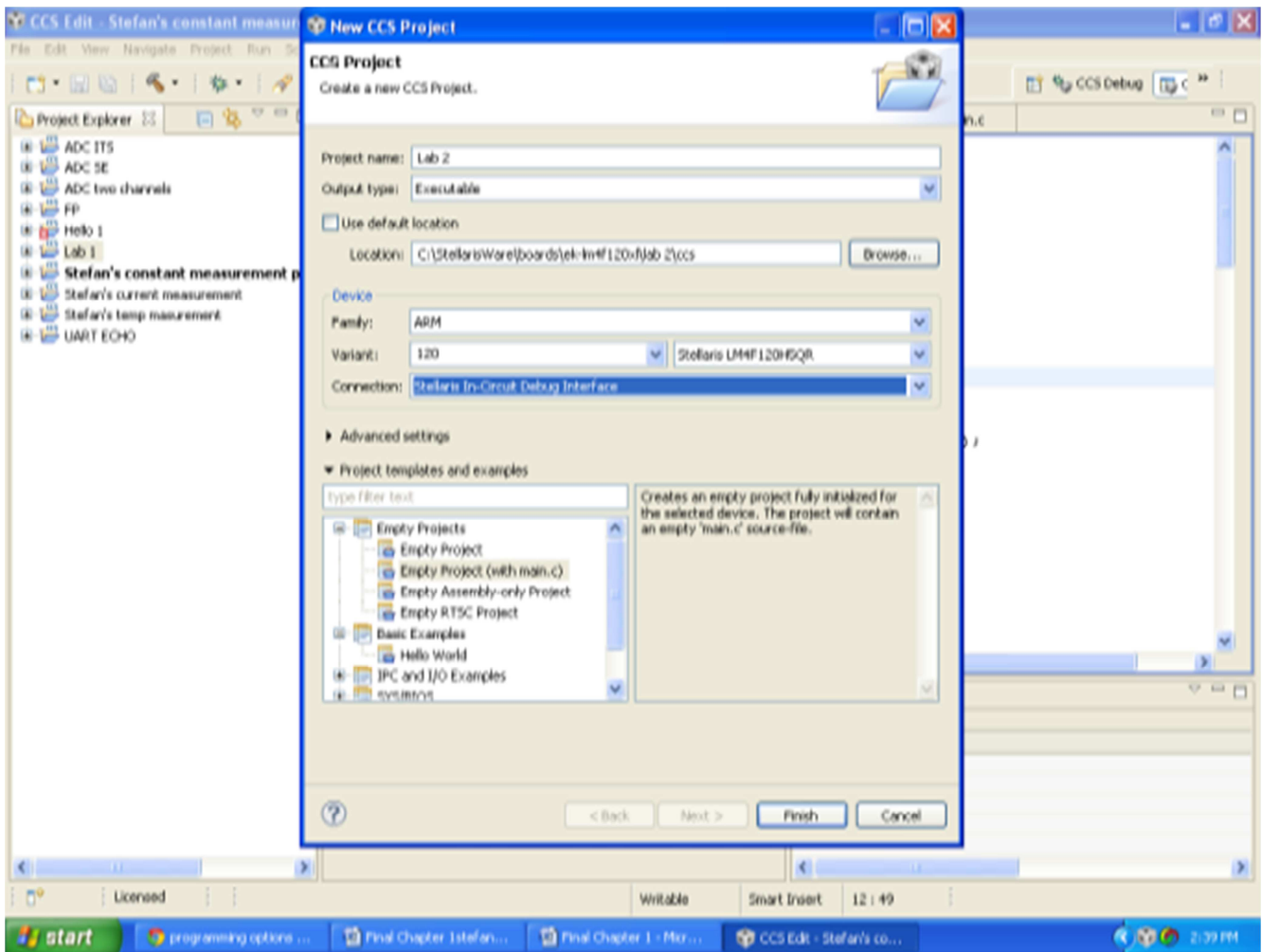


Figure 3. Creating a new project window (courtesy: Texas Instruments, USA).



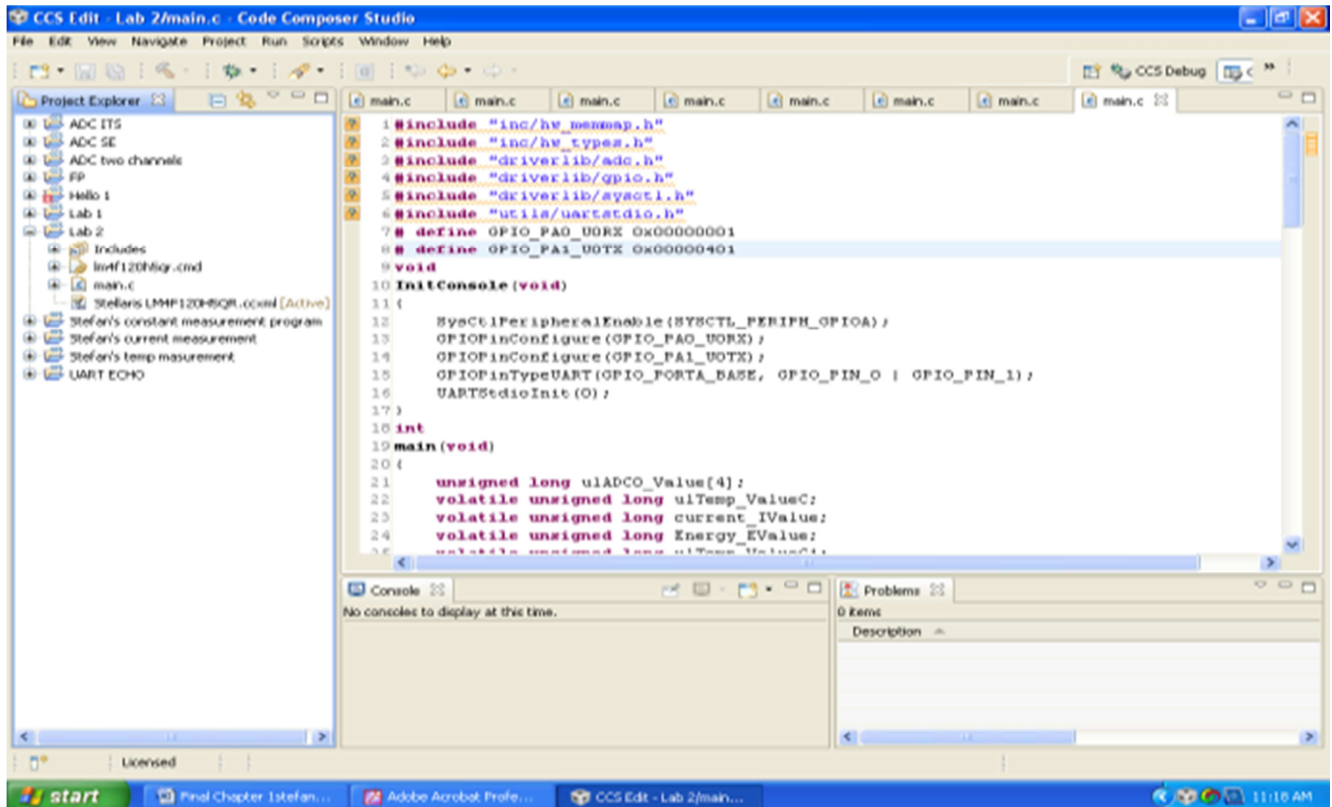


Figure 4. Saving C file with LED\_Blink (courtesy: Texas Instruments, USA).

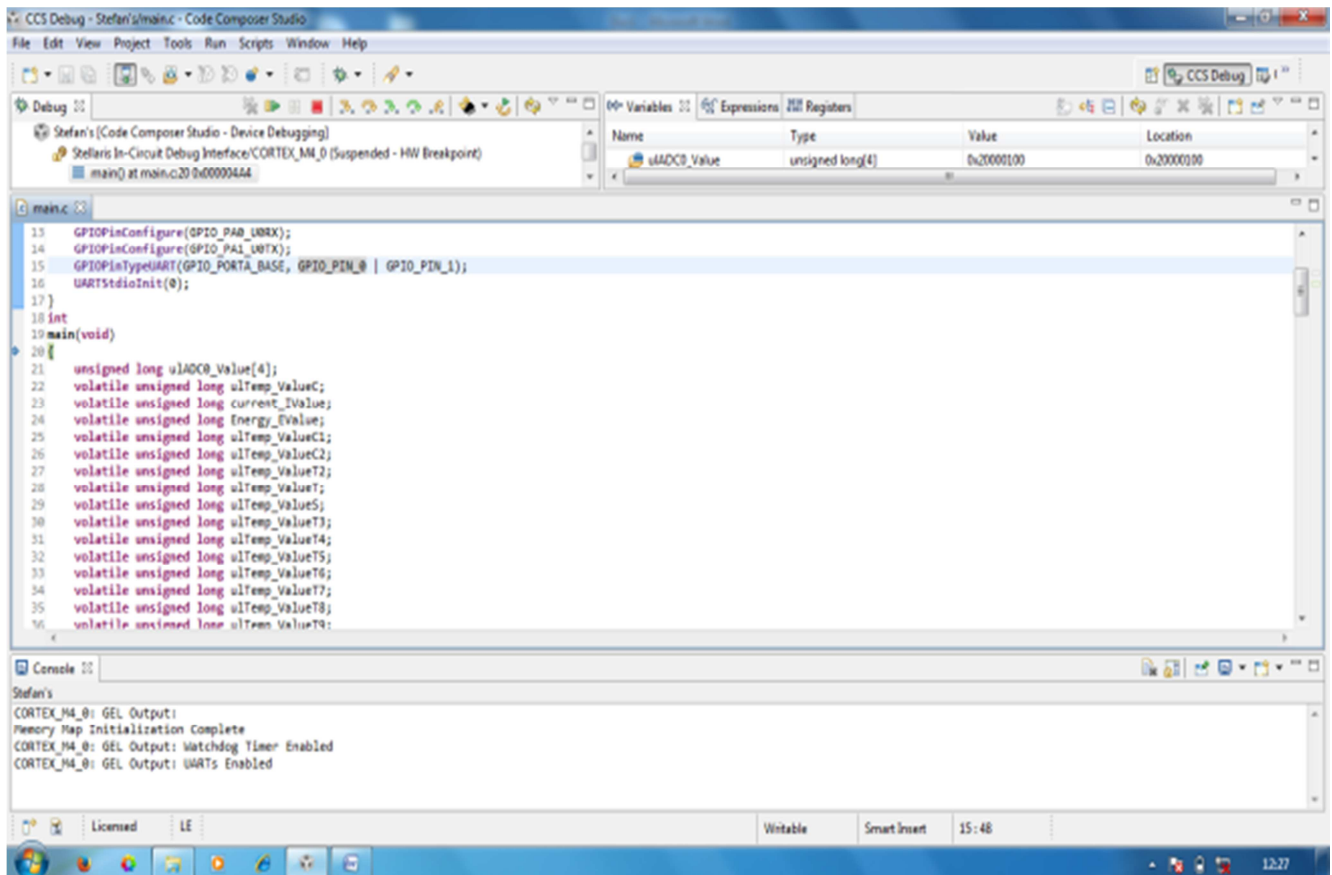


Figure 5. Debugging and running the LED\_Blink program (courtesy: Texas Instruments, USA).

## 4. Details of Individual Exercises

The total exercises developed and described in this paper are divided into three modules: A, B and C. The first module explains the basic input/output interfacing. Module B corresponds to the exercises for exploitation of the on-chip

peripherals (partially). Temperature measurement using a copper-constantan thermocouple interfaced with Stellaris LaunchPad is described in module C. Photograph of the Stellaris Launchpad with the on-board components is shown in figure 6.

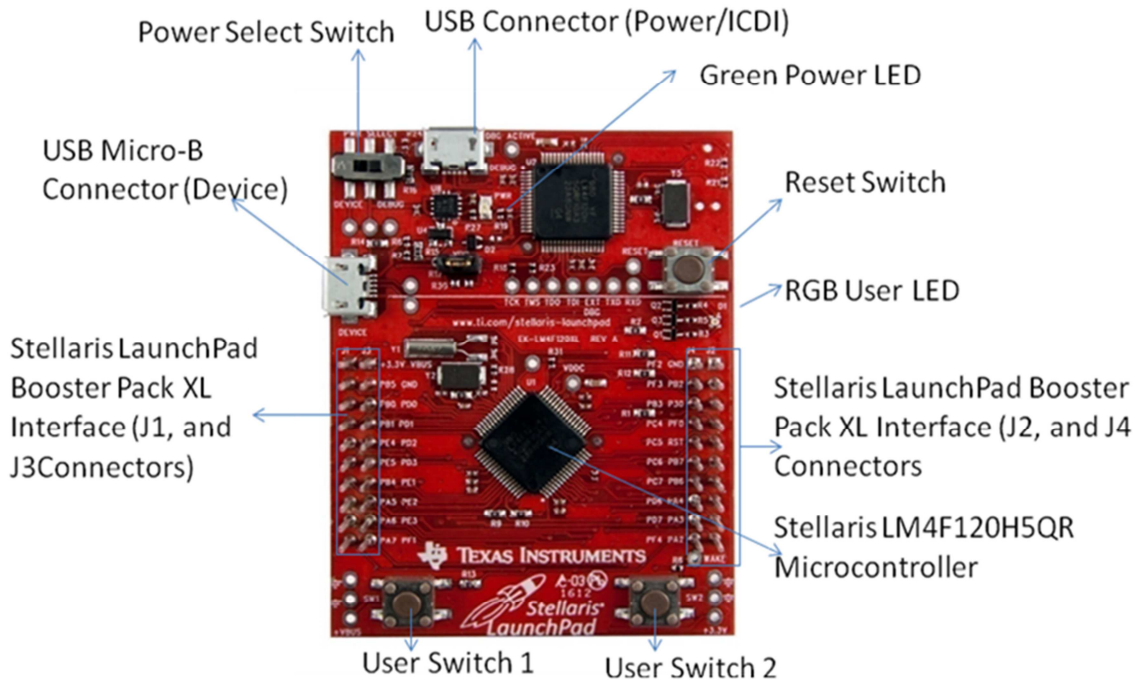


Figure 6. Stellaris LaunchPad development board (courtesy: Texas Instruments, USA).

### Module A: Basic I/O Interfacing

This module contains interfacing exercises, relating to I/O programming. Interfacing exercises starts with LED blinking program, seven segment, push button, four digit seven segment multiplexed display, stepper motor, and LCD display.

#### Ex. A 1: LED Blinking

The first exercise is LED toggling. This is the basic experiment in embedded systems. LED is the most commonly used component, usually for displaying the digital status of I/O pins.

To get acquainted with GPIO pins of the controller, LEDs are connected to port pins PB0 to PB 7 of PORTB with current limiting resistors (220Ω), mounted on a bread board. The program is developed in embedded C using CCS.

The software starts with first enabling the clock to the port by setting the register RCGCGPIO to 02. Declaring the GPIO port as output port, setting the drive strength and pull-up for each of the pins, enabling GPIO pins as digital I/O s are done by setting the registers: GPIODIR, GPIODR8R, GPIOPUR, and GPIODEN to FF. Now, alternate pins of PORTB are made high followed by a delay. Next, other four pins of PORTB are made high followed by a delay. The last two steps are kept in an infinite loop; as a result, alternate LEDs of PORTB blinks.

The outcome of this basic experiment is to understand the I/O pin characteristics of the microcontroller and connecting

the LED in positive or negative logic through the current limiting resistors.

Figure A1 shows the photograph of this exercise.

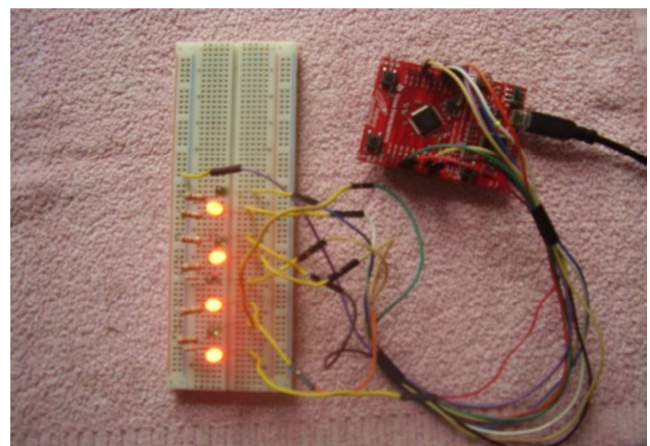


Figure A1. Stellaris LaunchPad interfaced with LEDs.

#### Ex. A 2: Seven segment Display

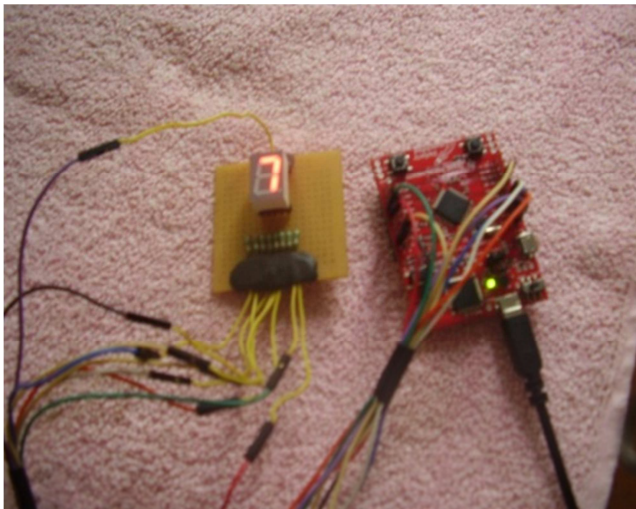
This exercise explains the interfacing of a common anode type seven segment display with PORTB to display hexadecimal numbers 0 through F. A seven segment display with current limiting resistors is assembled on a bread board



with eight wires for connecting to PORTB.

Software initially enables the GPIO pins as digital I/O by setting the registers mentioned in the previous exercise to FF, and setting PORTB as output. An array of sixteen elements to hold the HEX data corresponding to numbers 0 through F is initialized. Contents of array are sent to PORTB sequentially with delay. This program is kept in a loop. Figure A2 shows the seven segment displaying number 7 during the process of execution.

The outcome of the experiment is that the student will come to know about two types of displays: common anode, common cathode, and its usage in display circuits. As for as the software is concerned, student learns the initialization of arrays in embedded C using CCS.



**Figure A2.** Stellaris LaunchPad interfaced with seven segment display., Hex value '7' is displayed on the seven segment display.

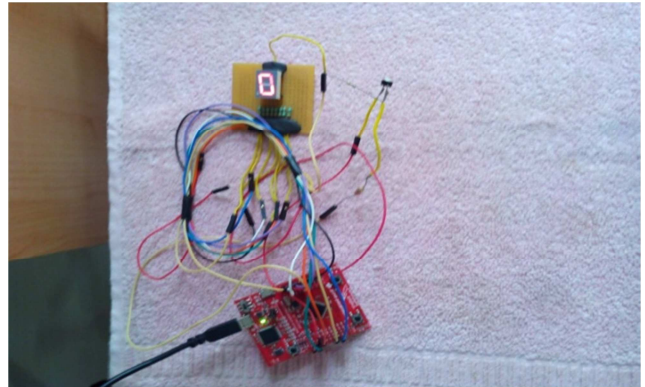
#### *Ex. A 3: Debouncing a pushbutton*

Mechanical switches have a problem called bouncing in the electronic domain, especially when used as a input to the microprocessor/microcontroller. Bouncing can be eliminated by either hardware or software. Hardware uses cross coupled NAND gates (basic RS flip-flop) or IC MAX6816, while the software needs programming (delay). In this exercise software debouncing is done by connecting a push button to PD2 of PORTD. A seven segment display connected to PORTB shows the number of times the button is pressed. Hardware is developed on a bread board by connecting pushbutton to PORTD and seven segment display to PORTB.

Program starts by loading the registers GPIODIR, GPIODR8R, GPIOPUR, GPIODEN to FF, which sets the direction of PORTB as output, drive strength and pull-up for each pin, enables the GPIO pins as digital I/O. GPIO register of PORTD is loaded with FB, since the pushbutton is connected to PD2. The count which reflects the number of times the button is pushed/pressed is displayed on the seven segment display.

The outcome of the experiment is that the student learns the necessity for debouncing a mechanical switch in general and the method of debouncing the button through software, in

particular. Figure A3 shows the pushbutton connected on a bread board and the display showing count 0 at the beginning.



**Figure A3.** Stellaris LaunchPad interfaced with seven segment display and push button.

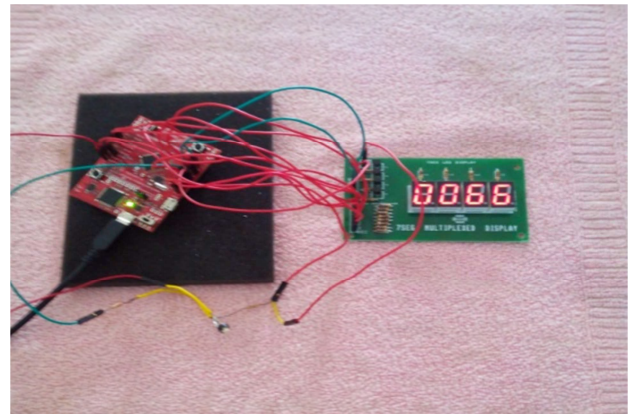
#### *Ex. A 4: Four-digit multiplexed seven segment displays*

In exercise A 3, a push button is interfaced with PORTD2, and the number of times the button is pressed is displayed on the single digit display. If the number of pushes crosses nine, then the display starts once again from zero. This problem is overcome in this exercise using a four-digit multiplexed display.

This exercise will serve two purposes, namely, the true count of the button pressing is displayed and multiplexing of display is implemented that results in average current drawn by the displays with less intensity/brightness.

All eight digit pins of each of the four seven segment displays are made common and connected to PORTB (PB0-PB7). Selection of one among four digits is accomplished using four pins of PORTE. Initialization of ports is done accordingly, through software. Further, the main software is developed for multiplexed display with the same debounce program explained in exercise A 2. A four-digit module available commercially is used in this exercise. Figure A4 shows the photograph of the interface.

The outcome of the experiment is that the student learns about what is multiplexing? How to multiplex displays What are the advantages of multiplexing seven segment displays?



**Figure A4.** Stellaris LaunchPad interfaced with seven segment multiplexed display.

*Ex. A 5: Stepper Motor Interface*

Motors can be broadly classified into two categories: ac motors, dc motors. Further, dc motors can be classified into two types: continuous rotation motors, stepper motors. Nevertheless, all motors are mechanical devices and are power hungry. Hence, if one wants to drive a stepper motor with microcontroller, he should be careful with the power driving capability of the port pins to which it is connected. It is always a good idea to use a buffer (current amplifier such as Darlington pair) between the microcontroller pins and the motor winding. In the present work a dc stepper motor working at 5V is taken. Its winding resistance is 15Ω. We have chosen ULN2003 as the power driver.

Winding of the stepper motor are connected to PORTB via ULN2003. The IC ULN2003 is placed on the bread board and jumpers are connected between the port pins and the motor. As the name indicates, stepper motor rotates in steps. The stepping angle per step varies from one motor to another motor. Motor stepping angle and speed can be controlled by software.

Software initialization is same as in exercise A 1. REGISTERB is initialized with 88H, and rotated right/left each time outputting the contents of REGISTERB, with delay in between. This rotates the shaft of the motor clockwise/anticlockwise. Figure A5 shows the photograph of this exercise.

Outcome of this exercise include the understanding of the working of stepper motor which is a mechanical device (having inertia), and its interfacing with microcontroller. Understanding its I/O pins driving capability (current sourcing and sinking) of ports I/O pins is an additional advantage.



**Figure A5.** Stellaris LaunchPad interfaced with stepper motor.

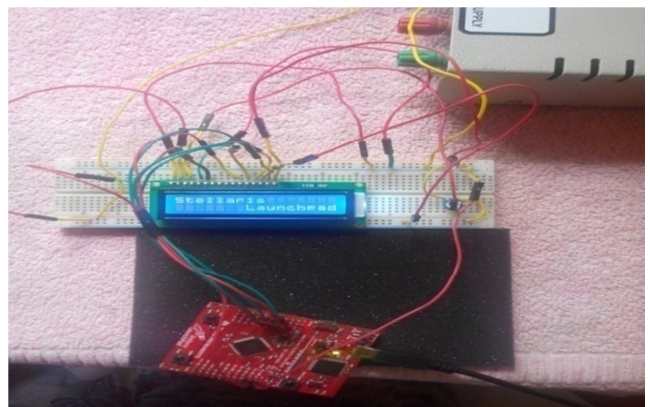
*Ex. A 6: LCD Interface*

Liquid Crystal Display (LCD) has the greatest advantage of low power consumption and occupies less space compared to the seven segment display used in the previous exercises (A 3 and A 4). Of course, it has a couple of major draw backs like the display appears clearly at a particular angle with a small distance, and needs more commands in software for its initialization. In the present work, a 16X2 LCD display is used for displaying characters [13]. A 10kΩ trim pot is used for brightness control of the backlight LED of the LCD module. The module is mounted on a bread board and connection between the LaunchPad pins and display are accomplished using jumper wires.

LCD module can be used in two modes: 8-bit mode and 4-bit mode. The 4-bit is also known as economical mode. In the present study, 4-bit mode is selected. PORTB is initialized as output port. RB4 to RB7 of PORTB pins are used for sending data to LCD, and RB2, RB3 are used as LCD control pins.

Besides setting the register for digital I/O, LCD commands are sent to choose 16 characters, 2 lines. Program contains two arrays: one consisting of characters: "STELLARIS" another array: "LAUNCHPAD". The data in the first and second arrays are output on first and second lines of LCD, respectively. Figure A6 shows the photograph of this exercise

From this exercise, the student learns the LCD functioning and its pros and cons compared with LED display. Total schematic diagram related to module A is shown in figure 7.



**Figure A6.** Stellaris LaunchPad interfaced with LCD.

Exploitation of on-chip peripherals is discussed in the following module.

**Module B: Interfacing Experiments to exploit the on-chip Peripherals**

This module gives the information about interfacing of the on-chip peripherals (partially) like Analog to Digital converter (ADC), Pulse Width Modulation (PWM), Universal Asynchronous Receivers/Transmitter (UART), and Inter Integrated Circuit (I2C).

*Ex. B1: UART Echo Exercise*

The microcontroller LM4F120H5QR has eight on-chip UARTs: UART0 to UART7. Baud rate is selectable. In the present exercise, UART0 is chosen with a baud rate of 115200 with 8-N-1 protocol.

The software needs the inclusion of library functions like: memmap.h, types.h, systcl.h, uart.h, gpio.h. All the peripherals related to UART0 are enabled, which includes the clock. The GPIO pinmuxing is configured for UART function (PORTA0 corresponds to RX and PORTA1 corresponds to TX). Program is developed to transmit the character typed on the keyboard serially on TX/RX lines to echo it on the hyperterminal of the laptop. The program is kept in a closed loop. The characters typed on the keyboard are displayed on the monitor of the laptop.

The outcome of this exercise include the selection and programming of the on-chip UART for serial data transmission to any peripheral connected to the



microcontroller over TX and RX lines. Figure B1 shows the photograph of this exercise.

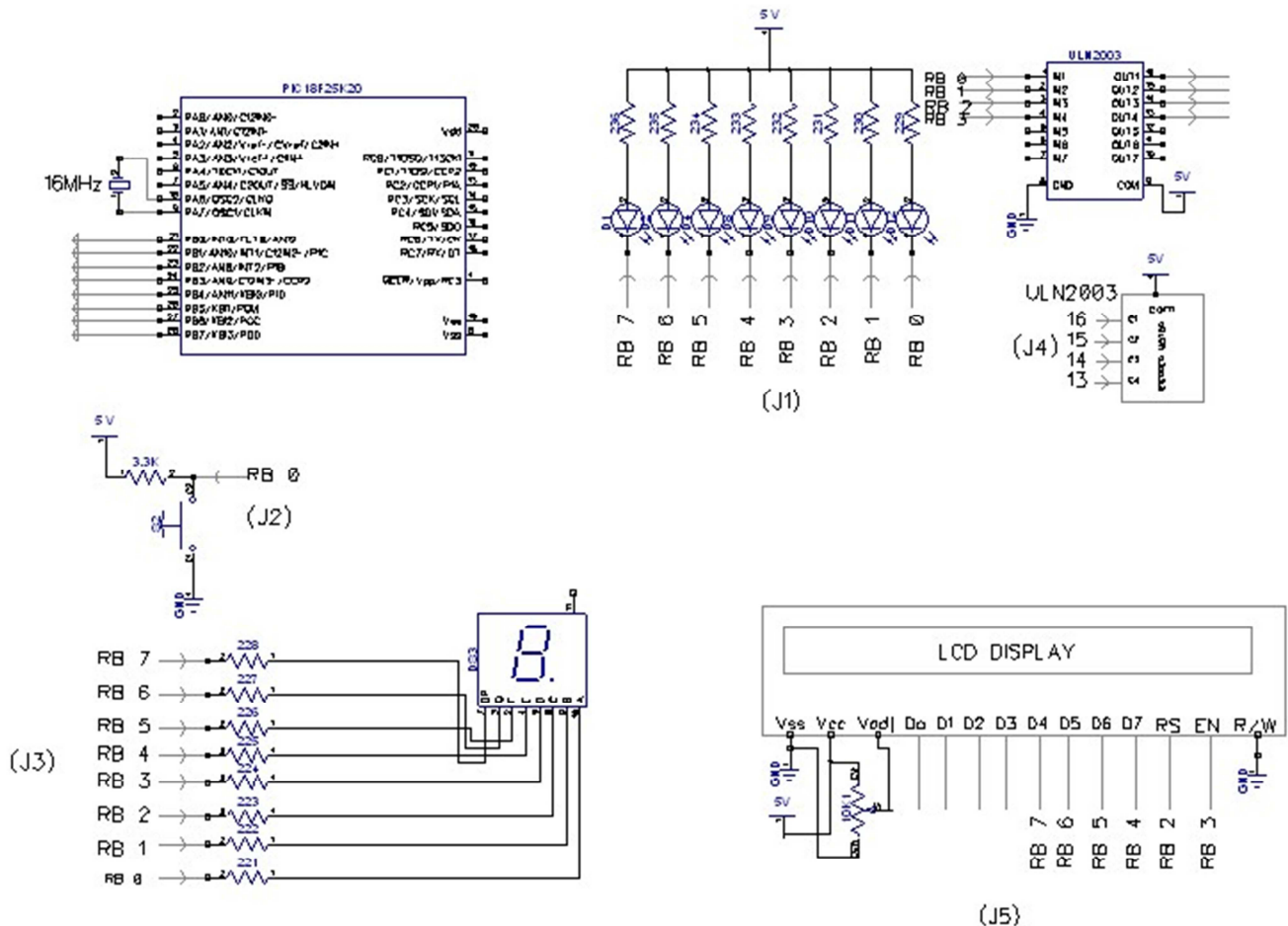


Figure 7. Total hardware for experiments A1 to A6 (Module A).



Figure B1. Photograph of UART Echo.

#### Ex. B2: Measurement of Analog Voltage using on-Chip ADC

The microcontroller used in the present work has two on-chip ADCs, each having twelve analog input channels. In the present exercise, ADC0 with analog input channel AN0 is used. The converted digital data equivalent to the analog input is sent to the laptop using the on-chip UART (Ex.B1).

The software starts with the inclusion of all required files like: memmap.h, types.h, adc.h, gpio.h, sysctl.h, uartstudio.h. UART0 is chosen to establish a serial link with the laptop in order to display the digital data. ADC0 is initialized for a single-ended and for a single sample. Sampling of the data is

done continuously and the data stored in the register ADCDATA is read and sent to laptop through UART. The digital value is displayed on the hyperterminal of the laptop.

The outcome of this exercise is that the student understands about the important ADC signals like: start-of-conversion (SOC), end-of-conversion (EOC), conversion time, ADC selection, input analog channel selection, clock selection for ADC. These are in addition to the serial transmission of ADC data to laptop with the knowledge of exercise B1. Figure B2 shows the photograph of this exercise.



Figure B2. Photograph of ADC Interfacing.

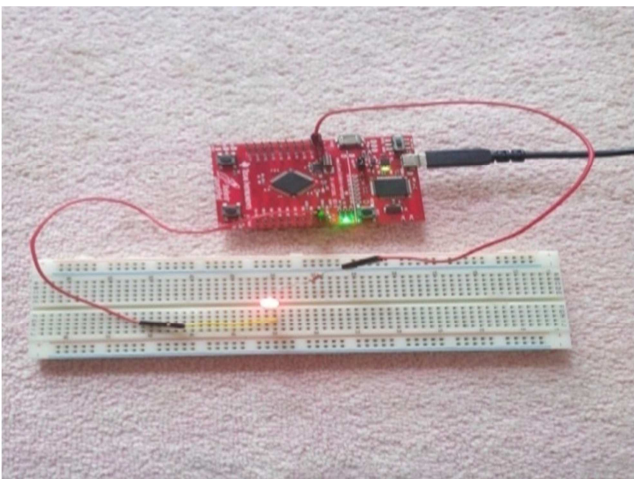
*Ex. B3: Realization of PWM using LED*

The microcontroller has twelve capture compare (CCP) PWM channels. They can be used in 16/32/64-bit modes. In the present work, CCP0 is used for producing PWM. Modulated output is obtained at PORTB6. An LED is connected to PORTB6 through a 150Ω resistor to observe the PWM. The same can be connected to the CRO in order to observe the waveform.

The software includes all files mentioned in the above exercises with an additional file:timer.h. Variables are defined that determines the duty cycle of the PWM pulses. The CCP0 peripheral is configured appropriately and enabled. The corresponding timer: TIMER0 is also enabled and configured as 16-bit periodic timer. It is loaded with 1000 and that of timer match is loaded with 250. Program with these inputs produces a PWM at PB6.

The outcome of this exercise is that the student gets to know about the various functionalities of the on-chip timer in general and production of PWM signals using CCP0 in particular.

Figure B 3 shows the photograph of this exercise



**Figure B3.** Photograph of PWM interfacing.

*Ex. B4: I2C (Inter-Integrated Circuit)*

In general I2C bus provides bi-directional data transfer through a two-wire design. This communication doesn't need level converters as in serial transmission using UART. The microcontroller presents four I2C modules (I2C0-I2C3) on the chip itself. The present exercise uses I2C0 module. The I2C module contains a master and a slave. Master and slave can be used in loopback mode to transfer data between master and slave by giving a common clock. In the present experiment, a loop back mode is accomplished besides transmitting the data to hyperterminal of the laptop using the on-chip UART0. This avoids interfacing with an external device having I2C pins.

The software starts with the inclusion of header files including the files that supports I2C and UART. I2C0, UART0 and clock modules are enabled. The GPIO PINMUX is configured for both I2C and UART functions (PORTB2 - I2C0SCL, PORTB3 - I2C0SDA; PORTA0 - UART0RX, PORTA1 - UART0TX). As a matter of fact, the Stellaris

microcontroller used here has an additional feature of I2C loopback mode that is useful for debugging I2C operation through software. This mode internally connects the I2C0 master and slave terminals, so that data sent by master can be received by the slave. The software can select the data transfer rate. If the verification of I2C operation has to be done by interfacing with external peripheral that supports I2C communication, then, PORTB2 and PORTB3 pins have to be connected to Vcc through external pull-up resistors.

An array of data is initialized, and the SLAVE ADDRESS is selected. Data is placed in DATA REGISTER. The RECEIVE BUFFER and DATA DIRECTION REGISTERS are initialized. After sending each data, the program waits for an acknowledgement from the slave. The data received by the slave is displayed on the hyperterminal of the laptop. After resetting the receive buffer, the direction of data transfer is changed so that the master reads the data that is sent from the data buffer of the slave. This is done after a dummy read (receive) by the master so that junk data will be avoided. After dummy acknowledge, the program waits for a send request from the master to the slave. Data read by the master is also sent to the laptop's hyperterminal. Thus the data communicated between master-to-slave and vice versa on I2C0 lines can be verified physically on the hyperterminal.

Figure B4 shows the photograph of this exercise



**Figure B4.** Photograph of I2C module interfacing.

*Module C: Sensor Interfacing with Stellaris LaunchPad (LM4F120H5QR)**Ex. C1: Temperature measurement using copper-constantan thermocouple sensor interfaced with microcontroller*

Temperature measurement is an attractive exercise which involves sensor interfacing. There are different types of temperature sensors available on the market, like thermocouple, semiconductor (pn-junction sensors) and IC sensors.

In the present study, copper-constantan thermocouple is used as a temperature sensor. As the output of the thermocouple is in the range of millivolt, it is necessary to amplify the signal before interfacing with microcontroller. Analog Device's AD595 thermocouple amplifier is used to boost the output of the thermocouple [14]. Output from the



thermocouple is filtered for noise signal before connecting it to the amplifier. Figure 8 shows the schematic of this exercise.

The software starts with all header files mentioned in exercise B2. UART0 and ADC0 are used in the present exercise to display the temperature and to convert the analog output of AD595 to digital data, respectively. Analog input channel AN0 is fed from the output of the AD595. After conversion of the analog voltage equivalent to the equivalent digital value, a look up table converts this digital value to an equivalent temperature in °C. This value is displayed on the hyperterminal of the UART. Figure C1 shows the photograph of this exercise.

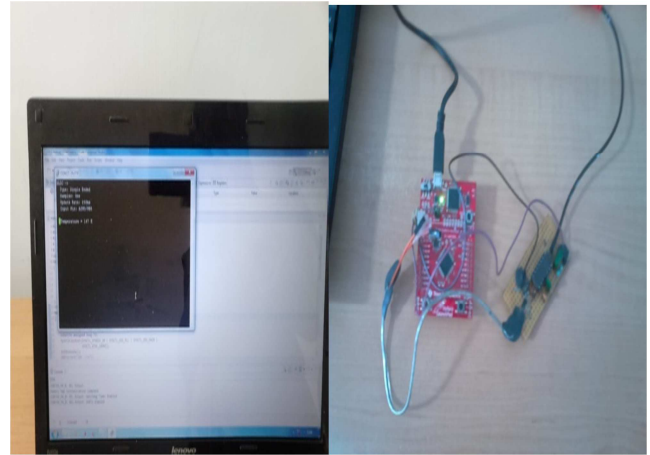


Figure C1. Photograph of temperature sensor interfacing with Stellaris LaunchPad.

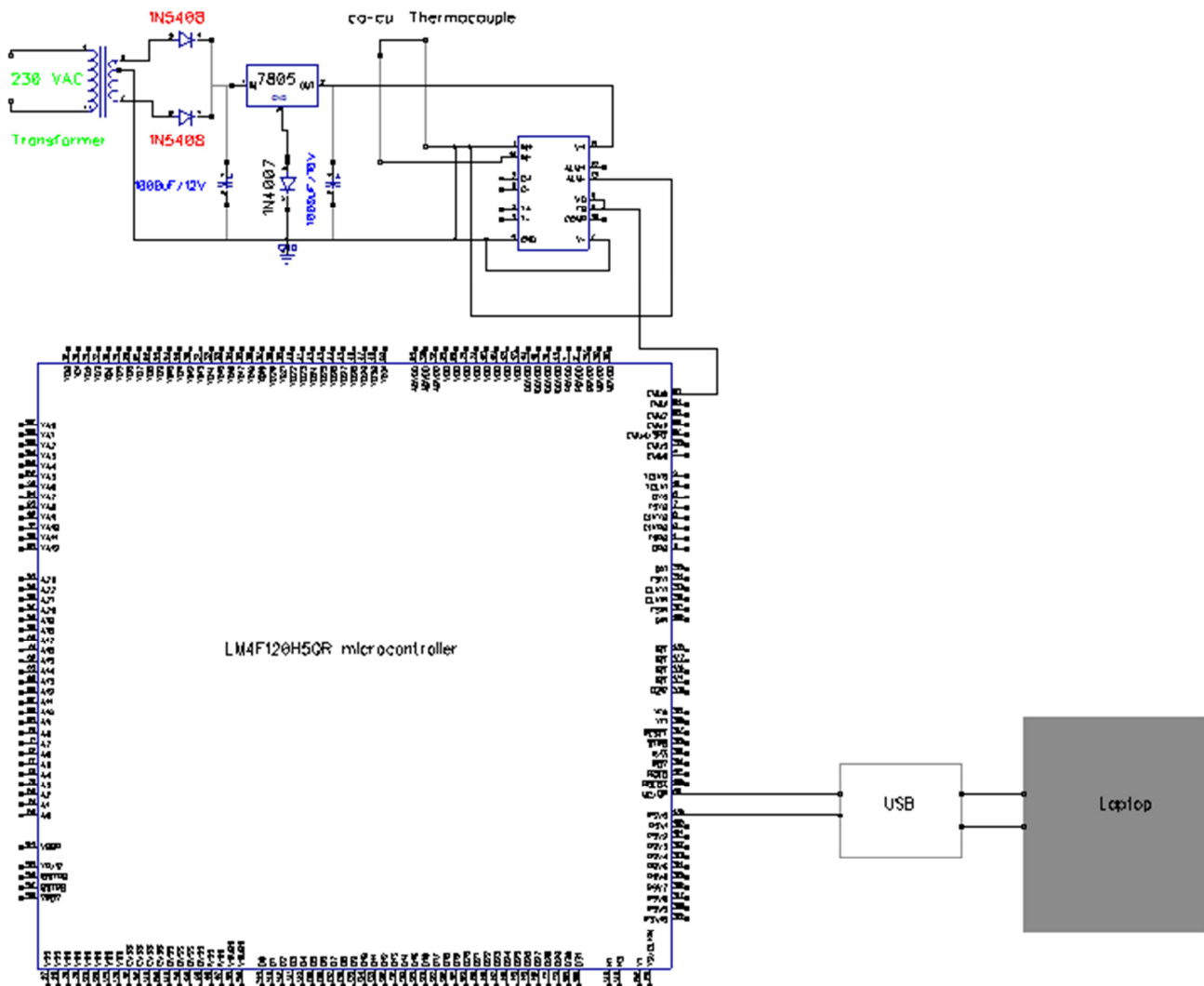


Figure 8. Hardware Schematic of AD595 thermocouple amplifier.

## 5. Student Feedback and Survey

We got the feedback from the students by asking them to fill up

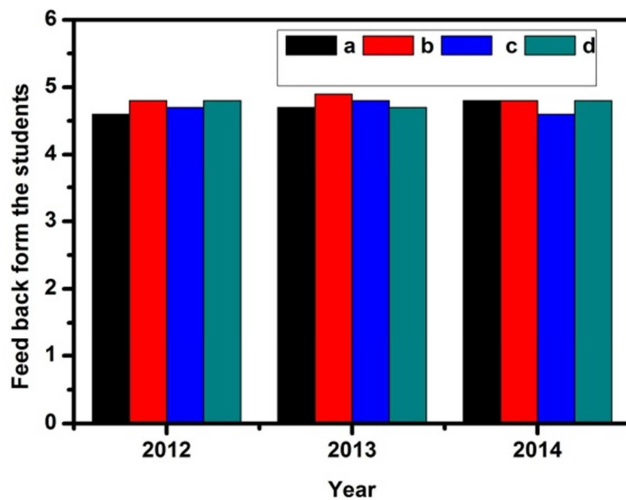
a questionnaire. The results of the feedback are shown in Table 1.

Student's response over a length of three years is shown in Figure 9.

**Table 1.** The Average scores of the survey for exercises.

Questions	Score (1-5)		
	2012	2013	2014
(a). What was your understanding about the lecture on ARM cortex-M4 based system design?	4.6	4.7	4.8
(b). Are you satisfied with the ARM Cortex-M4 (TIs' Stellaris LM4F120H5QR) lab kit?	4.8	4.9	4.8
(c). What was your understanding about hardware interfacing and software development?	4.7	4.8	4.6
(d). Are you satisfied with the hands-on experience and the type of sensor based exercise?	4.8	4.7	4.8

1-poor, 5-Excellent

**Figure 9.** Students response over a length of three years.

## 6. Conclusion

In this paper, we have designed and developed certain exercises to get acquainted with the microcontroller LM4F120H5QR using the Texas Instruments' Stellaris Launch Pad board. The Launch Pad presents majority of the pins of the microcontroller brought to the on-board connectors. The exercises relating to the Input/Output interfacing, exploitation of on-chip peripherals such as: ADC, UART, PWM, I2C and a thermocouple interface for temperature measurement are described. Still there are un attempted on-chip peripherals: SSI and USB interface. Successful attempt of these exercises with advanced projects will increase the confidence in the student to think and develop embedded system design, which is essential in the present scenario.

## Acknowledgements

The facility created by the Department of Science and Technology (DST), New Delhi by sanctioning the FIST program (Phase-I) is acknowledged.

C. Rajeswari is thankful to University Grants Commission (U.G.C), New Delhi, for sanctioning of Junior Research Fellowship (B.S.R).

Software for the above programs can be obtained from author on request.

## References

- [1] David Lawrence, Didier Buchs, and Armin Wellig, "Using In instrumentation for Quality Assessment of Resilient software in Embedded systems" I. Majzik and M. Vieira (Eds): SERENE2014, LNCS 8785, PP 139-158, 2014.
- [2] Sheng Hongyu, and Wei Gaung, on the "Embedded Training System project", the idea [J]. Laboratory research and exploration, 2005, 24 (supplement):60-64.
- [3] Pang Ling, An Lei, "The discussion about teaching of embedded system experiments", IEEE, 2010.
- [4] Li Tu, Jun Yang, "Research on Experimental Teaching of Embedded Systems", International conference on Education Technology and Management Engineering, vol.02, pp 16-17, 2012.
- [5] Naveen Kumar Uttarkar and K. Raghavendra Rao "Design and Development of a Low-Cost Embedded System Laboratory Using TI MSP430 Launch Pad", American Journal of Embedded Systems and Applications. Vol. 1, No. 2, pp. 37-45, 2013.
- [6] Aruna. Kommu and Raghavendra Rao Kanchi "Design and Development of Project based Embedded System laboratory using LPC1768" American Journal of Embedded systems and Applications, vol. 1, No. 2, Dec2013, pp 46-53.
- [7] Swapna Chintakunta, Raghavendra Rao Kanchi and Ramanjappa Thogata, "Designing an Introductory FPGA-based Embedded System Laboratory", American Journal of Embedded Systems and Applications. Vol.2, No.2, pp.6-12, March, 2014.
- [8] Rajeswari Cherukuri and Raghavendra Rao Kanchi "Design and development of a project-based embedded system laboratory using PIC 18F25K20", American Journal of Embedded Systems and Application, Vol. 2, No. 3, June 2014, pp.21-28.
- [9] Datasheet, <http://www.ti.com/ds/2/405/lm4f120h5qr-12404.pdf>.
- [10] User Manual, <http://www.ti.com/lit/ug/spmu289c/spmu289c.pdf>.
- [11] Workbook, [http://www.ti.com/Stellaris\\_Launchpad\\_start\\_files/Stellarislaunchpadworkbook.pdf](http://www.ti.com/Stellaris_Launchpad_start_files/Stellarislaunchpadworkbook.pdf).
- [12] <http://www.ti.com/tool/ccstudio>.
- [13] <http://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-5V.pdf>.
- [14] [http://www.analog/media/en/technical\\_documentation/data\\_sheets/AD594-595.pdf](http://www.analog/media/en/technical_documentation/data_sheets/AD594-595.pdf).